CRC 알고리즘

- **CRC** 알고리즘을 이해한다.
- **CRC** 알고리즘을 이용한 데이터 무결성을 구현할 수 있다.

1) 패리티 비트
2) Check Sum
3) CRC 알고리즘
4) CRC TABLE 구현 및 확인

Error Detection

◆ 정리

- 데이터 전송이 데이터값이 변경되는 경우가 발생
- 데이터가 올바로 전송됐는지 확인하기 위한 값 필요
  - 체크섬값을 이용
- 체크섬값
  - 각각의 데이타값을 더한 값
- 전송값
  - < data><checksum>
- Checksum값도 변경되는 경우가 발생할 수 있다.
- 데이터 값들이 변경되었는데도 checksum값과 같은경우
  - Checksum 오류에도 보정 가능한 기능의 함수 필요

패리티 비트(parity bit)

```
char data = 'a';

if( hweight(data)%2 )
   data |= 0x80;
```

```
   01100001
   10000000 |
   ----------
   11100001
```

```
char data = 'a';

if( hweight(data)%2 == 0 )
   올바른 데이터
else
   변형된 데이터
```

```
   11101001
```

패리티 비트(parity bit)

패리티 비트(parity bit): 한계점 -> 짝수개의 비트가 변형되면
error를 찾을 수 없다.

```
char data = 'a';

if( hweight(data)%2 )
   data |= 0x80;
```

```
01100001
10000000 |
----------
11100001
```

```
char data = 'a';

if( hweight(data)%2 == 0 )
   올바른 데이터
else
   변형된 데이터
```

```
11101001
```

Check Sum

송신부

```
struct iphdr {
    __u8    ihl:4,
            version:4;
    __u8    tos;
    __be16  tot_len;
    __be16  id;
    __be16  frag_off;
    __u8    ttl;
    __u8    protocol;
    __sum16 check;
    __be32  saddr;
    __be32  daddr;
};
```

```
45
10
00 c8
24 4c
40 00
40
06
00 00
c0 a8 38 80
c0 a8 38 01
```

```
  4510
  00c8
  244c
  4000
  4006
  0000
  c0a8
  3880
  c0a8
+ 3801
--------
 2DBFB
```

```
2DBFB
 DBFB
    2 +
-----
 DBFD

~ 1101 1011 1111 1101
  0010 0100 0000 0010
    2   4     0  2
```

Check Sum

```
struct iphdr {
    __u8    ihl:4,
            version:4;
    __u8    tos;
    __be16  tot_len;
    __be16  id;
    __be16  frag_off;
    __u8    ttl;
    __u8    protocol;
    __sum16 check;
    __be32  saddr;
    __be32  daddr;
};
```

수신부

```
45
10
00 c8
24 4c
40 00
40
06
24 02
c0 a8 38 80
c0 a8 38 01
```

```
  4510
  00c8
  244c
  4000
  4006
  2402
  c0a8
  3880
  c0a8
+ 3801
--------
 2FFFD
```

```
 2FFFD
  FFFD
     2 +
-----
  FFFF

~ 1111111111111111
  0000000000000000
    0   0   0   0
```

check sum의 한계점
: 데이터중 하나는 더하기 되고 하나는 빼기가 되는 경우
전체의 합에는 변동이 없으므로 error를 발견할 수 없다.

CRC(Cyclic Redundancy Check)

송신부

```
11010011101100 |
1011           |
 1100011101100 |
 1011          |
  111011101100 |
  1011         |
   10111101100 |
   1011        |
       1101100 |
       1011    |
        110100 |
        1011   |
         11000 |
         1011  |
          1110 |
          1011 |
           101 | 0
           101 | 1
               | 100
               |
               |
```

수신부

```
11010011101100 | 100
1011           |
 1100011101100 | 100
 1011          |
  111011101100 | 100
  1011         |
   10111101100 | 100
   1011        |
       1101100 | 100
       1011    |
        110100 | 100
        1011   |
         11000 | 100
         1011  |
          1110 | 100
          1011 |
           101 | 100
           101 | 100
               | 000
               |
               |
```

CRC(Cyclic Redundancy Check)

```
data = 0x12;
crc  = 0xac
```

송신부

```
00010100 | 10101100
  111010101
   10010101
   111010101
    11111111
    111010101
        10101100
```

수신부

```
00010100 | 10101100
  111010101
   10011111100
   111010101
    11101010100
    111010101
        00000000
```

CRC(Cyclic Redundancy Check)

```
data[3] = {0x12,0x34,0x0};
crc  = crc8(0x2d^0x34)
data[2] = crc = 0xae;
```

```
 00101101
 00110100 ^
 00011001
```

송신부

```
00011001 | 10101110
   11101 0101
     1000101
     111010101
      11000001
      111010101
         10101110
```

수신부

```
00011001 10101110
   11101 0101
     10011111110
     111010101
      1110101010
      111010101
         00000000
```